REMARKS

Applicant has amended claims 9, 10, 11, 27, and 35 to further clarify features of embodiments of the present invention. Applicant submits that no new matter has been entered by these amendments. Similarly, Applicant submits that the amendments previously submitted in Amendment B also do not introduce new matter, as will be explained below.

Claims 9-11 and 22-25 stand rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the written description requirement. Particularly, the Examiner states that the phrase in claim 9, "independent of a particular code of the computer program" is not found in the specification. Applicant has amended claim 9 to clarify each of the plurality of intervals of execution is selected independently of particular lines of code of the computer program. Claims 10 and 11, formerly depending from claim 9, have been amended to depend from claim 1, and should, along with their dependent claims 22-25, be removed from the rejection. With respect to claim 9 as amended and newly amended claims 27 and 35, the amendments find clear support in the application as originally filed.

The application makes clear that intervals of execution, or arbitrary sections of execution, are selected independently of particular code. Page 11, lines 6-9 make clear that an interval is a selection of continuous instructions in program execution order, and may be thought of as a slice of time. An example of an interval (page 21, lines 18-23) is N X 100 million instructions. Note that this definition of an interval does not state that particular instructions are run (i.e., an interval is not defined as the instructions identified, for example, at lines 150 million – 350 million), only that N X 100 million instructions are run within this interval.

As further stated on page 11, lines 13-15, a statistic is tracked for a component, and the statistic is correlated with the component over the interval of execution. The statistic correlated with the component thus depends on the particular interval of execution chosen.

An example component provided in the application (page 11, lines 15-17; page 14, lines 19-20) is a program component such as an identifiable section of control flow of the computer program, such as a basic block of the code. An example statistic for this component provided in the application (page 12, lines 1-2; page 18, lines 10-15) is a frequency of a basic block over an interval of instructions. A basic block (page 20, lines 8-9) is a section of code that is executed from start to finish with one entry and one exit. The interval of execution is thus the independent variable in this analysis, and the frequency of the basic block is the dependent variable, because the frequency of the basic block depends on the particular interval.

This feature is further supported by one or more figures in the application and their accompanying description. FIGs. 4A-4F show, for each of a plurality of intervals on an x-axis (also known as the independent axis), distances between the behavior of that particular interval and that of a target interval, where the target interval is based on the behavior of the entire program (page 21, line 4 – page 24, line 12). This behavior is determined by calculating a frequency that each of a number of basic blocks has been entered within the particular interval. The distances, a dependent variable, are shown on the y-axis (otherwise known as the dependent axis).

Thus, in this example, a statistic measuring the particular code being run (the number of times a basic block is entered) is dependent on the particular interval selected, <u>not the other way around</u>. This is also why the interval is shown on the x-axis and the statistic is shown on the y-axis, as would be understood by one of ordinary skill in the art. Similarly, FIGs. 5A-5F, 6A-6F, 7A-7C, and 15-20 show an x-axis (aka independent axis) of intervals, with a y-axis or axes (aka dependent axis or axes) showing various behaviors of the program as it is executed. For example, FIGs. 7A-7C clearly show intervals as the independent variable (on the x-axis), and the behavior of the intervals based on particular building block durations as dependent variables. FIGs. 15 and 18 show a comparison of various intervals based on their behavior, as determined by tracking the basic blocks executed during the intervals (page 41, line 5 – page 42, line 11).

In all of these cases, the interval is selected independently of the particular lines of code that are run. On the other hand, the particular code being run is an example behavior that is dependent on the interval selected. Thus, it is not new matter to claim that each of the plurality of intervals is selected independently of particular lines of code. The application as filed, including the drawings, clearly supports this feature, as would be understood by those of ordinary skill in the art.

Please also note that this approach is opposite to the approach taken by Goodnow. Goodnow consistently selects particular lines of code as the independent variable (input). This is not surprising, as the purpose of Goodnow is to evaluate particular lines of code so that segments of code can be selected for debugging or for copying to another program. Goodnow does not teach copying lines of code regardless of their content. In Goodnow, the results that are analyzed are provided by selecting particular lines of code, whether they are blocks, functions, or code segments, and determining their static and dynamic attributes.

Claims 1-8, 12-17, 26-33, 35-40, 42-48, 50, and 52 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Goodnow. Applicant respectfully traverses the rejection for at least the reasons submitted in Amendment B. Additionally, with respect to the above-identified differences between embodiments of the present invention and Goodnow, Applicant will address each of the Examiner's points raised in the "Response to Arguments" section of the Office Action. The Examiner's remarks will be addressed in order, except for the remarks related to definitions of certain terms (remarks (C) and (D)).

In remark (D), the Office Action cites the present specification, page 11, lines 6-12 for the definition of "interval", but apparently ignores this definition because multiple examples are given. An interval, as clearly set out in the specification, is a selection of continuous instructions in program execution order (claim 11 has been further amended to directly incorporate this definition). While the specification states that an interval may also be thought of as a slice of time, this further explanation does not contradict the given definition. For example, one can run instructions for a period of time, and the number of continuous instructions run in program execution order during that time, would be the interval.

Similarly, the cited portion of the specification goes on to state that the interval may be a time interval, an instruction interval, and/or a metric-based interval. However, none of these examples contradicts the previously-provided definition, nor do the examples provide a justification to ignore this definition. Particularly, whether the interval is selected by a particular amount of time, a particular number of instructions, or a particular metric (e.g., executing a program until a certain metric is reached), the interval still may be considered a "slice of time", and is still a selection of continuous instructions in program execution order. Note that <u>no part of this definition</u> states that particular lines of code are selected as lines of code and then run.

Regarding the claim term "arbitrary sections of execution", the Office Action (remark (D)) cites a dictionary definition of "arbitrary", but then fails to apply the definition provided. Looking at the very portion of the definition bolded in the Office Action (page 6), "arbitrary" is defined as "based on or determined by individual preference or convenience <u>rather than by</u> necessity or <u>the intrinsic nature of something</u>" (emphasis added). The sections of executions according to embodiments of the present invention are clearly "arbitrary" under this definition as they are not based on or determined by the intrinsic nature of the code itself: the particular lines of code.

These definitions are not applicable to the code segments found in Goodnow, as such code segments are based on or determined by their intrinsic nature; that is, their particular lines of code. The Examiner (remark (C)) cites col. 4, lines 41-46 of Goodnow for defining code segments, but this definition is incomplete in view of the remainder of Goodnow's written description. To clarify, Goodnow defines "blocks" as a single entry such as a line instruction (col. 9, lines 13-14), and a "function" as a particular collection of blocks (col. 10, lines 16-17). Goodnow defines two "code segments" as two functions within either the same program or different programs or any other designation such as, but not limited to, blocks, Lvalues (i.e., line number values), or statements (i.e., statements of code). Examples of code segments and functions (functions each represent a code segment – col. 4, lines 62-65) are shown in TABLE 1 (col. 5, lines 1-9), and each of these is defined by specific lines of code (lines 18-26; lines 56-65; etc.)

It is important to note that all of these definitions in Goodnow, whether for blocks, functions, or code segments, whether the code segments be in a single program or different programs, whether they be designated by blocks, Lvalues, or statements, have at least one thing in common: **they are all defined by, and restricted to, particular lines of code**. Again, the purpose of Goodnow's invention is to determine static and dynamic attributes of such code segments so that they can be debugged or used in other programs. "Code segments", "blocks", and "functions", under any definition provided in Goodnow, are not independent of particular lines of code, nor are they based on something other than their intrinsic value. Thus, they are not intervals of execution, nor are they arbitrary sections of execution as defined in the pending claims.

Noting the differences between intervals of execution and arbitrary intervals of execution as in embodiments of the present invention, on the one hand, and code segments (or blocks, or functions) in Goodnow, on the other hand, Applicant will now address the remaining remarks in order. Regarding remark (A), the Examiner cites col. 4, lines 39-41 and col. 11, lines 50-54 of Goodnow for purportedly teaching the claimed comparing identified behavior for at least one interval of execution to another interval of execution to determine similarity. The cited section does teach similarity measurements, but between code segments, not intervals of execution (see also col., 3, lines 14-21). As explained above, an interval of execution is independent of the particular lines of code. However, in Goodnow, a code segment is directly dependent on particular lines of code, as that is how a code segment is defined (see TABLE 1).

The Office Action also appears to interpret a single function in Goodnow to be equivalent to a plurality of intervals. This is incorrect, as such functions are not a plurality of arbitrary sections of execution, nor are they a plurality of intervals of execution as defined. Additionally, the Office Action's analysis appears to interchange Goodnow's treatment of blocks and that of code segments when comparing the reference to the pending claims, which contradicts the teachings of Goodnow. As one of several examples, the Office Action's analysis re claim 1 cites a function in Goodnow as teaching that a data structure "identifies each function (plurality of intervals)" (page 8) for the claimed features of running code over a plurality of intervals of execution, thus appearing to equate blocks and intervals, but then alleges (page 9) that "a data structure identifies each block (of an interval/function)" for the claimed features of comparing behavior of one interval of execution to another, thus equating a block, an interval, and a function. Separately (page 9), the Office Action cites col. 11, lines 50-54 for producing a distance measurement indicating similarities between identifiers and function types within code segments being compared, thus also equating code segments with intervals of execution.

Regarding remark (B), the Office Action cites Goodnow, col. 8, lines 62-66 for teaching the claimed identifying behavior of a hardware-independent metric within at least one arbitrary section of execution of a portion of a computer program and classifying each of the at least one arbitrary section of execution according to the identified behavior into clusters of similar behavior. The cited section of Goodnow teaches that dynamic attributes of code segments may be extracted (versus static attributes, which are determined before the code segment is run). Again, the code segments are not arbitrary sections of execution (nor are they intervals of execution). Further, cited TABLE 6 and col. 9, lines 24-26 teach identifying what blocks, when executed, transition to themselves or to other blocks, not comparing code segments. While col. 2, line 60 appears to teach generating a cluster interface, this interface is for code segments, not blocks. The Office Action's analysis makes it unclear as to whether blocks or code segments are being cited as the claimed intervals of execution or arbitrary sections of execution (though, again, neither qualify based on the definitions above).

Regarding remark (C), the Office Action now appears to be relying on blocks as representing the claimed intervals of execution, and cites TABLE 6 for teaching identifying behavior of a hardware-independent metric for each of a plurality of intervals and comparing the identified behavior of each of the plurality of intervals to an identified target behavior. Again, blocks in Goodnow cannot be the claimed intervals of execution, as explained above. Further, the Office Action cites col. 13, lines 10-53 for teaching the claimed cluster interface comparing the identified behavior to the identified target behavior, but claim 35 requires that the target behavior be identified by identifying behavior of a hardware-independent metric of full execution of the at least a portion of the computer program. In the cluster interface described in col. 13, the similarities are mapped based on predefined constraints (col. 13, lines 18-22), not on a target behavior as claimed.

For at least these reasons, Applicant respectfully submits that claims 1-8, 12-17, 26-33, 35-40, 42-48, 50, and 52 are allowable over the references or record, including Goodnow. Applicant thus respectfully requests reconsideration and withdrawal of the rejection.

Applicant also wishes to point out the Examiner's obligation regarding patentable subject matter according to MPEP 706:

> When an application discloses patentable subject matter and it is apparent from the claims and the applicant's arguments that the claims are intended to be directed to such patentable subject matter, but the claims in their present form cannot be allowed because of defects in form or omission of a limitation, the examiner should not stop with a bare objection or rejection of the claims. The examiner's action should be constructive in nature and when possible should offer a definite suggestion for correction...If the examiner is satisfied after the search has been completed that patentable subject matter has been disclosed and the record indicates that the applicant intends to claim such subject matter, he or she may note in the Office action that certain aspects or features of the patentable invention have not been claimed

and that if properly claimed such claims may be given favorable consideration.

Applicant has particularly pointed out clear differences between embodiments of the present invention and the teachings of Goodnow. If the Examiner appreciates these differences but believes the claims still do not reflect such differences, Applicant respectfully requests suggestions and notes as set out in MPEP 706, in the interest of expediting prosecution.

Claims 9-11 and 22-25 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Goodnow in view of Moreno. Applicant respectfully traverses the rejection for at least the reason that neither Goodnow nor Moreno teaches or suggest that each of a plurality of intervals of execution is independent of particular code of the computer program, nor that each of the plurality of intervals comprises at least one of a time interval in which instructions are run, a number of instruction executed, and an interval of instructions run having a length determined by a code independent metric. The Examiner cites Goodnow for teaching that a code segment source 105 provides segments of code, representative of one or more programs, for statistical analysis. However, these code segments provided are not independent of particular code of the computer program, but instead they are based on selecting particular lines of the computer program for analysis. Examples of such generated code segments are shown in TABLE 1 for the program shown in col. 3-col. 4. These code segments are particularly identified lines of code; that is, they are based on the selection of particular lines of code (as opposed to, for example, the selection of the first 100 lines run during execution, the second 100 lines run during execution, etc). Though the code source may encompass a wide variety of devices and processes, the selection of code source is always a specific line or lines as identified by the code itself.

The Examiner recognizes that Goodnow fails to teach or suggest that segments may be based on a time interval or a number of instructions. Moreno is cited to remedy this deficiency, but this secondary reference is unrelated to Goodnow. The Office Action states that it would have been obvious to one of ordinary skill in the art to modify Goodnow in view of Moreno, because "timed interval sampling was fairly known in the art, as a method to reduce sample collection, while still representing the overall program." The interval sampling in Moreno is directed to streams of a received transmitted multimedia signal ("information stream"), not to code generation (a program). There is nothing in Moreno that applies the idea of sampling multimedia signals to selecting intervals of execution during running code.

Goodnow is specifically and directly focused on analyzing specific pieces of code, not arbitrary pieces, and not pieces generated as a result of an interval of execution. Nothing in Goodnow contradicts this, and nothing in Moreno teaches or suggests that this fundamental feature of Goodnow should, or could, be modified. Even if the specific pieces of code are extracted by a code source, they are selected as particular lines of code to be analyzed both statically and dynamically.

Claim 34 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over Goodnow in view of Clark. Claims 18-21, 41, 49, 51, and 53 stand rejected under 35 U.S.C.

§ 103(a) as being unpatentable over Goodnow in view of Clark and further in view of Baker. Applicant respectfully traverses the rejection for at least the reasons stated above regarding Goodnow, and for at least the reason that Clark and Baker fail to remedy the deficiencies of Goodnow.

For at least the foregoing reasons, Applicant believes that this case is in condition for allowance, which is respectfully requested. The Examiner should call Applicant's attorney if an interview would expedite prosecution.

Respectfully submitted,

GREER, BURNS & CRAIN, LTD.

By: [illegible]

Arik B. Ranson
Registration No. 43,874

Customer No. 24978
January 24, 2008
300 South Wacker Drive
Suite 2500
Chicago, Illinois 60606
Telephone: (312) 360-0080
Facsimile: (312) 360-9315